# Java Tutorium

Thomas Krieger, Satz: Helge Janicke

3. März 2001

## Inhaltsverzeichnis

# 1 Java Tutorium Zusammenfassung

# 2 Begriffe in Java

## 2.1 Java

Java ist eine unter vielen Programmiersprachen. Sie sticht allerdings aus verschiedenen Gründen etwas hervor. Java ist eine Objektorientiert Progammiersprache. Programmierer können einmal geschriebene Objekte ständig wiederverwenden, weiter vererben und erweitern. Dadurch kann auch durch einen relativ geringen Programmieraufwand schon ein anständiges Programm entstehen. Ausserdem können Objekte nicht nur in einem speziellen Programm sondern in jedem beliebigen Programm benutzt werden. Auch eine grosse Besonderheit von Java ist, das jedes in Java geschriebenes Programm im Gegensatz zu vielen anderen Programmiersprachen auf jedem Betriebssystem ausgeführt werden können. Einzige Voraussetzung ist ein installiertes Java. Es gibt noch weitere Besonderheiten, die allerdings nicht so zur Last fielen.

## 2.2 Compiler

Der Javacompiler dient zum übersetzen der Quelldateien (s. weiter unten) in ein für den Computer verständliche Sprache. Er ist sozusagen ein Dolmetscher von Programmierer zum Computer. Der Compiler wird unter Windows im DOS-Fenster aufgerufen oder unter Unix im Terminal. Verschiedene Programmierumgebungen rufen den Javacompiler auch direkt auf. Der Aufruf sieht unter beiden Betriebssystemen gleich aus :

```
c:\>javac Test.java      bzw.    bash2.03:#>javac Test.java
```

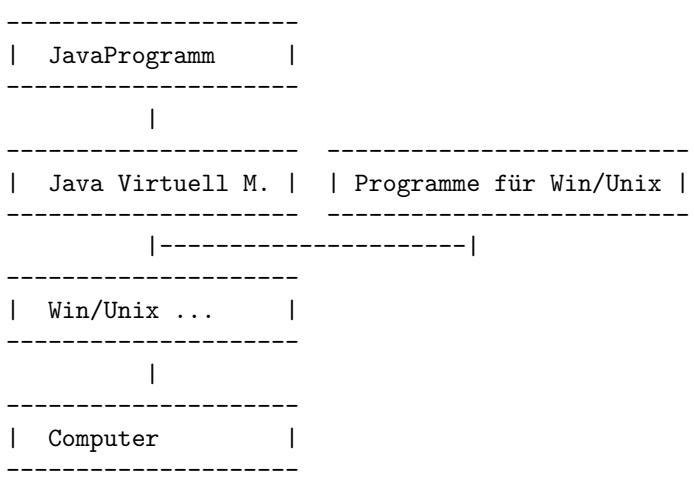
Dabei wird zu erst das Programm javac und dann die zu übersetzende Javadatei oder auch Quelldatei (erkennbar durch den Dateianhang .JAVA) angegeben. Sollte der Compiler beim übersetzen auf einen Fehler trefen, so bricht er den Vorgang ab und gibt eine Fehlermeldung aus. Dabei wird zu erst die Datei genannt, in der der Fehler aufgetreten ist, danach die Zeile in der der Fehler gefunden wurde und als letztes eine Beschreibung des Fehlers.

TIP :

Sollten einmal mehr als ein Fehler vom Compiler ausgegeben werden, so sollte man zuerst den oberen Fehler suchen und beheben. Danach kann dann ein zweiter Versuch die Datei zu übersetzen schon Fehlerfrei gelingen, da die anderen Fehler häufig Folgefehler sind, die sich aus einer fehlenden Klammer oder einem fehlenden Semikolon hervorgerufen wurden.

## 2.3 JVM

Die Java Virtuell Machine ist ein simulierter PC im PC. Dies dient dazu um Java- programmen eine einheitliche Umgebung zu ermöglichen. Dadurch können Javaprogramme auf allen Betriebssystemen ausgeführt werden, auf denen eine JVM installiert ist. Man kann sich die JVM wie eine Schicht zwischen PC und Betriebssystem vorstellen.




## 2.4 Java Software Development Kit

Das Java SDK oder JDK ist ein komplettes Softwarepacket von der Firma SUN (java.sun.com) zum erstellen und ausführen von Javaprogrammen. Das Java SDK ist aus dem Internet frei zum herunterlagen verfügbar.

## 2.5 Java Runtime Enviroment

Die Java Runtime Enviroment enthällt unter anderem die JVM und ermöglicht so als installierbares Softwarepacket die Ausführung von Javaprogrammen. Die Java Runtime Enviroment (JRE) ist ein Teil des Java SDK um es jedem zu ermöglichen Javaprogramme auszuführen, auch ohne das komplette Java Software Development Kit auf dem HeimPC zu installieren.

## 2.6 Javainterpreter

Der Javainterpreter führt die programmierten Javaprogramme aus. Er verknüpft sozusagen das Javaprogramm (erkennbar durch den Dateianhang .CLASS) mit der JVM. Ausführbar sind allerdings nur .CLASS Dateien in denen eine sogenannte main-Methode vorhanden ist (siehe auch Methode oder Applikation).

## 2.7 Applikation

Eine Applikation ist ein selbstständig ausführbares Programm. In Java sind Applikationen dadurch erkennbar, das sie eine Methode mit Namens main in sich haben. Die Main-Methode ist sozusagen der Startpunkt von jedem Javaprogramm. Eine .CLASS Datei ohne main-Methode kann daher nie alleine ausgeführt werden. Ein Applikation unter Java ist daher eine .CLASS Datei mit einer main-Methode.

## 2.8 Quelldatei

Die Datei in der der P̈rogrammtextv̈om Programmierer geschrieben wird. Also der eigentlich vom Menschen lesbare Teil um das Programm zu schreiben. Unter Java haben Quelldateien die Dateiendung .JAVA. Jede Quelldatei spiegelt eine Klasse wieder und muss den entsprechenden Namen der Klasse auch

als Dateiname verwenden. Klasse : Auto Quelldatei : Auto.java Wichtig dabei ist wie überall in Java ist auch hier auf Gross- und Kleinschreibung zu achten.

## 2.9 Vererbung

Vererbung bedeutet unter Java soviel wie alles das was die Urklasse hat, bekommt auch die neue Klasse. Jede Methode der alten Klasse kann genauso in der Neue Klasse verwendet werden. Dadurch können Programmierer für jeden Bereich eine Grundklasse erstellen und aus dieser dann alle speziall Fälle erstellen. z. B. aus der Klasse Lebewesen → Mensch, Hund, Katze...

## 2.10 Aggregation

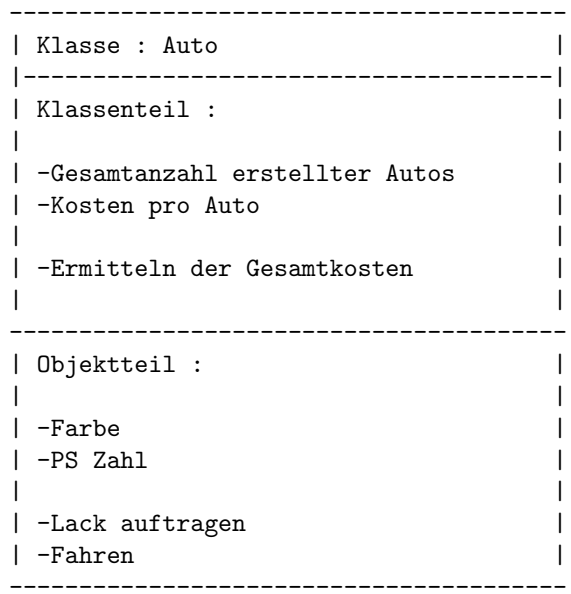
Aggregation kommt den Vererben sehr nahe. Dabei werden auch wieder alle Methoden und Attribute der Urklasse an die neue Klasse übergeben. Jedoch heist Aggregation mehr die Erstellung einer neuen Klasse aus mehreren Klassen. z. B. aus den Klassen Motor, Rahmen, Sitze ... → Auto

## 2.11 Objekt/Instanz

Da Java eine objektorientierte Programmiersprache ist, handelt sich vieles Beim Programmieren um die sogenannten Objekte. Ein Objekt ist eine Programmdatei (.CLASS) die ein im Programm benutzbares Objekt wiederspiegelt. In einem Programm für eine Autofirma kann z.B. ein Auto oder eine Karosserie ein Objekt darstellen. Oder in einem Programm für die Fachhochschule könnte ein Student oder ein Professor ein Objekt darstellen.

## 2.12 Klasse

Eine Klasse ist der Bauplan für ein Objekt unter Java. Daher beschränkt sich die Programmierung unter Java hauptsächlich auf das erstellen und verknüpfen von Klassen. Jede Klasse wird durch eine .JAVA also Quelldatei (siehe weiter oben) dargestellt. In dieser Datei können alle Eigenschaften der Klasse bestimmt werden. Aus einer Klasse können beliebig viele Objekte erstellt werden. Ein Klasse kann aber auch als eigenständiger Teil des Programms gesehen werden. Dieser Teil kann auch Informationen speichern und Aufgaben erledigen wie jedes Objekt auch. z. B. :




## 2.13 Datentypen

Datentypen spiegeln bestimmte Behälterformen für Informationen da. z.B.

- int
  kann Ganzzahlen aufnehmen.
- double
  kann Gleitkommazahlen aufnehmen.
- String
  kann Text aufnehmen.

auch Klassen sind Datentypen. Sollte ein Datentyp für bestimmte Informationen nicht geeignet sein, so gehen diese meist beim übergeben verloren z.B. die Nachkommastellen von double nach int.

## 2.14 Attribute

Attribute sind Eigenschaften einer Klasse oder eines Objektes die als Werte zurückgeliefert werden können. Wie in dem Beispiel oben ist z.B. die Farbe ein Attribut des Objektes Auto. Jedes Attribut hat einen Datentyp und einen Namen um darauf zugreifen zu können.

## 2.15 Methoden

Methoden sind Teile einer Klasse, die benutzt werden, um einen Arbeitsvorgang, der dem Objekt oder der Klasse zugeordnet ist als ein Teil zusammen zu fassen. Sie spiegeln sozusagen das Können der Klasse oder Objekt wieder. Also alles das was die Klasse oder das Objekt tun kann. Wie in dem Beispiel obe ist z.B. fahren eine Methode des Objektes Auto. Jede Methode hat einen Namen. Die meisten Methoden liefern auch ein Ergebniss zurück den sogenannten Rückgabewert der Methode.

### 2.16 Konstruktor

Der Konstruktor ist eine spezielle Methode, die jedesmal beim erstellen eines Objektes aus der Klasse ausgeführt wird. Dabei dient der Konstruktor zum Initalisieren der Attribute und anderen Dingen. Der Konstruktor kann genauso wie jede andere Methode Parameter erwarten.

### 2.17 Überladen

Sollte es nötig sein, eine Methode in mehreren Varianten zu haben, so kann man diese überladen. D. h. es werden mehrere Methoden mit gleichen Namen aber unterschiedlichen Parametern und / oder Rückgabewerten festgelegt. Dabei kann jede Methode eine Analoge funktion den anderen gegenüber übernehmen. Die überladenden Methoden werden anhand der Parameter und des Rückgabewertes unterschieden. Da der Name ja gleich ist.

### 2.18 Variablen

Variablen sind Behälter für Informationen, die eine Methode benötigt. Jede Variable hat einen Datentyp und einen Namen um auf Sie zugreifen zu können. Variablen werden z.B. in Schleifen als Laufvariable benutzt.

## 3 Syntax

Syntax sind sozusagen die Regeln, die man beim schreiben von Programmen beachten muss.

### 3.1 Semikolon

Jede Anweisung wird durch ein Symikolon beendet.

```
x = y * 100;
```

### 3.2 Zuweisungen

Um einer Variablen oder einem Attribut einen Wert zu zuweisen, benutzt man ein einzelnes = Zeichen.

```
x = 5;   oder   y = 5 * a;
```

### 3.3 Vergleiche

Um zwei Werte miteinander zu vergleichen, werden 2 = Zeichen benutzt also ==. Jeder Vergleich spiegelt einen booleschen Wert wieder also True oder False. Dabei steht True für Ja, beide Werte stimmen überein und False für Nein beide Werte haben einen unterschiedlichen Wert.
( x ist gleich 5 und y gleich 3)

```
x == 3; // gibt false wieder, da 5 ungleich 3
x == 3 + 2; // gibt true wieder, da 5 genau das gleiche ist, wie 3 + 2
x == y; // gibt false wieder,.......
```

## 3.4 Methodenaufbau

Methoden gehören immer einer Klasse an. Demnach werden sie auch immer in einer Klasse reingeschrieben. Eine Methodendeklaration enthält immer ein Zugriffsattribut, einen Rückgabewert und einen Namen. Sollte die Methode als Klassenmethode später benutzt werden, so muss nach dem Zugriffsattribut (z.B. public) das Schlüsselwort static angegeben werden.

```
public static double rechneWas() {
  return ergebnis;
}
```

- public
  Zugriffsattribut
- static
  Schlüsselwort für Klassenmethoden oder Attribute
- double
  Rückgabewert
- rechneWas
  frei erfundener Name der ganz kurz die Aufgabe der Methode wiederspiegelt

Der Rückgabewert kann jeden beliebigen Datentyp haben. Das Schlüsselwort static wird nur gebraucht, wenn es sich um eine Klassenmethode handelt. Wenn ein Rückgabewert angegeben ist, so muss ein Wert vom entsprechenden Datentyp über die return Anweisung zurückgegeben werden. Soll z.B. das Ergebnis einer Addition zurückgegeben werden :

```
public int addiere(int a, int b)
  {
  int erg;
  erg = a+b;
  return erg;
  }
```

Hier werden unter anderen auch Parameter benutzt. Parameter werden in den Klammern hinter dem Methodenname geschrieben. Dabei gilt immer zu erst der Datentyp und dann ein frei erfundener Name. Sollten mehrere Parameter von nöten sein, so werden diese durch Komma getrennt. Die Reihenfolge muss dann beim Aufruf der Methode berücksichtigt werden.

```
...
x = addiere(5,3);
...
```

a ist jetzt 5 und b ist 3. x wird dem Ergebnis also 8 zugewiesen. Die 8 wird durch return in Form von der Variable erg zurückgegeben.

## 3.5 Klassenaufbau

Um eine Klasse zu erstellen, sollte man sich als erstes einen Namen für diese Ausdenken, und eine Datei mit genau diesem gewünschten Namen erstellen + .java. z.B. Klasse Mensch → Mensch.java Die Klasse in der Datei fängt (vernachlässigen wir die Zugriffsattribute mal) immer mit public class und dem Klassennamen an.

```
public class Mensch {
  }
```

Alles was jetzt zu dieser Klasse gehört, wird hinter dem Namen in den geschwungenen Klammern gesetzt. Die geschwungenen Klammern grenzen also den Klasseninhalt von anderen Dingen ab. Sie schliessen die Klasse sozusagen in sich ein. Der normale Aufbau einer Klasse ist immer :

```
public class Mensch {

  // Attribute
  int GebJahr;

  // Methoden
  public double rechne(int a,int b) {
    ...
    return erg;
    }

  public void gehe() {
    ...
    }

  }
```

Möchtet ihr eine Klasse aus einer anderen erstellen, sprich vererben, so müsst ihr dies ebenfalls in der ersten Zeile der Klasse angeben.

```
public class Mensch extends Klassenname {
  }
```

Das Schlüsselwort extends mit dem gewünschten Klassennamen gibt an, welche Klasse hier weiter vererbt wird. z.B

```
public class Student extends Mensch {
  }
```

Dadurch erhält der Student alle Methoden und Attribute der Klasse Mensch.

## 3.6 Variablen-/Attributendeklaration

Variablen müssen, bevor sie benutzt werden, deklariert werden. Das heist Sie bekommen einen Namen und einen Datentyp. Dabei wird zuerst der Datentyp gegeben und danach ein Name der möglichst gut zu der Variable passt. z.B. eine Variable die zum durchzählen benötigt wird : int count; Attribute werden normalerweise direkt zum Anfang einer Klasse deklariert. z. B.

```
public class Auto {

//Attribute
int PS;
...
}
```

Variablen werden immer in der Methode deklariert, in der Sie benötigt werden. z.B.

```
public class Auto {
...
public void fahren()
  {
  String wohin;
  ...
  }
...
}
```

## 3.7 Die if - Anweisung

Die if Anweisung dient dazu einen bestimmten Teil des Programmes nur auszuführen, wenn eine bestimmte Bedingung erfüllt ist. Bei der if Anweisung kann falls gewünscht auch ein else Teil angefügt werden der im Falle der nicht erfüllten Bedingung ausgeführt wird. Die Bedingung wird immer in Klammern geschrieben. Der Auszuführende Teil wird wie üblich in geschwungenen Klammern eingefasst.

```
if (Bedingung)
  {
  Mach dies;
  }
```

oder auch :

```
if (Bedingung)
  {
  Mach dies;
  } else
  {
  Sonst halt das;
  }
```

z. B.

```
if (x<10)
  {
  System.out.println("Das Ergebnis ist kleiner als 10\n");
  } else
  {
  System.out.println("Das Ergebnis ist groesser oder gleich 10\n");
  }
```

## 3.8 Die for - Schleife

Die for - Schleife dient meist dazu, einen bestimmten Vorgang einer um eine bestimmte Anzahl zu Wiederholen. Sie besteht im ganzen aus drei Teilen 1. dem Initalisierungsteil 2. dem Bedingungsteil und 3. dem Anweisungteil. Der Initalisierungsteil dient dazu um eine Variable einen bestimmten Wert zu zuweisen. Der Bedingungsteil gibt an welche Bedingung erfüllt sein muss, damit die Schleife wiederholt wird. Solange die gegebene Bedingung Wahr ist wird die Schleife weiter ausgeführt. Der Ausführungsteil wird nach jedem durchlauf ausgeführt. Er wird meist dazu genutzt die im Initalisierungsteil initalisierte Variable zu inkrementieren ( um einen zu erhöhen ). Die Bedingung wird zum Anfang und nach jedem durchlauf des Anweisungsblockes geprüft.

```
for (Initalisierung;Bedingung;Anweisung)
  {
  zu Wiederholender Block von Anweisungen
  }
```

z.B.

```
for (i=0;i<100;i+=2)
  {
  sum += i;
  }
```

Das Beispiel durchläuft alle 2er Werte in der Variablen i und summiert sie in der Variablen sum.

## 3.9 Die while - Schleife

Die while - Schleife durchläuft einen bestimmten Anweisungsblock solange, solange eine gegebene Bedingung erfüllt ist. Die Bedingung wird zum Anfang und nach jedem durchlauf des Anweisungblockes geprüft.

```
while (Bedingung)
  {
  zu Wiederholender Block von Anweisungen
  }
```

z.B.

```
i=0;

while (i<100)
  {
  System.out.println(i + ". Zahl");
  i++;
  }
```

Diese while - Schleife durchläuft die Zahlen 0 - 99 und gibt für jede die Meldung
0. Zahl
1. Zahl
2. Zahl ...

aus. Dabei gehört das i=0; nicht zur Schleife. Jedoch ist eine initalisierung meist auch hier notwendig.

## 3.10 Die do while - Schleife

Die do while - Schleife ist ganz ähnlich der while - Schleife (2.9) der grosse Unterschied besteht hier in dem Zeitpunkt der ersten Prüfung der Bedingung. Während bei der while - Schleife die Bedingung auch zum Anfang geprüft wird, und daher der Anweisungsblock unter Umständen kein einziges mal ausgeführt wird. Die do while - Schleife prüft die Bedingung nur nach dem Ausführen des Anweisungblockes, d. h. der Anweisungsblock wird mindestens einmal ausgeführt!

```
do {
  zu Wiederholender Anweisungsblock
} while (Bedingung);
```

z.B.

```
do {
  System.out.println("Möchten Sie weiter machen (Ja/Nein) ? ");
  testString = Tastatur.liesString();
  } while ((testString.compareTo="Ja")==0 || (testString.compareTo="Nein")==0);
```

Diese do while Schleife liest solange einen String von der Tastatur, bis der Benutzer wirklich Ja oder Nein eingegeben hat. Das Objekt Tastatur ist nur als Beispiel benutzt worden und existiert nicht unter Java.

## 3.11 Objekte aus Klassen erstellen

Um aus einer von Java gegebenen Klasse oder auch einer eigenen Klasse eine Objekt zu erstellen, welches man in seinem Programm benutzen möchte, muss man den new - Operanden benutzen. Das Schlüsselwort new legt ein neues Objekt aus der angegebenen Klasse an und führt den gewünschten Konstruktor aus.

Klassenname Objektname = new Klassenname();

Der Klassenname bezeichnet die Klasse, aus der das Objekt erstellt werden soll. Der Objektname, ist ein frei erfundener Name, der das Objekt wiedergeben soll. Die Klammern hinter der 2. Angabe des

Klassennamens spiegelt wieder, das hier der Aufruf des Konstruktors statt findet. Dabei können auch je nach Klasse Parameter übergeben werden. z.B.

String text = new String("Test Text");

oder

Jabberwok jb = new Jabberwok();

Solltet ihr eine Klasse aus Java benutzen wollen, müsst ihr darauf achten, das diese dem Compiler bekannt ist. Unter Java werden alle Klassen in sozusagene Verzeichnisse strukturiert, um Sie einer bestimmten Kategorie unter zu Ordnen. Standard gemäß kennt Java nur alle Klassen des Verzeichnisses java.lang. Wie man sieht werden die Verzeichnisnamen nicht mittels / oder \ getrennt sondern durch den Punkt. Um jetzt eine Klasse aus Java zu benutzen, kann man in der Java Dokumentation sich diese Ansehen und am Anfang steht auch in welchem Javaverzeichniss sich diese befindet. z.B. java.io. Solltet ihr also eine Klasse aus java.io benutzen wollen, so gibt ihr in der Entsprechenden Klasse über dem public class ... eine Importanweisung an. z.B.

```
import java.io.*;

public class ...
```

Danach weiss der Compiler er soll nicht nur unter java.lang sehen sondern auch im Verzeichnis java.io alle (*) Klassen nach diese Klasse durchsuchen.

## 3.12 Methodenaufruf

Wenn ihr also jetzt eine Methode aus einem Objekt oder einer Klasse aufrufen wollt, müsst ihr zuerst wissen, wie sie heisst, und welche Parameter sie braucht. Unter Umständen braucht man auch den Rückgabewert, da man ja das Ergebnis der Methode benutzen möchten. Der Rückgabewert wird meist in einer Variablen vom gleichen Datentyp gespeichert.

```
Variable = Methodenname(Parameter mit Komma getrennt);
```

z.B.

```
int erg;

erg = Mathe.addiere(5,8);
```

Hier wird eine Variable vom Typ int (Integer) angelegt. Das Ergebnis der Methode addiere aus dem Objekt Mathe wird dann in diese Variable gespeichert. Die Methode könnte also so definiert sein :

```
public int addiere(int a, int b) {
...
}
```

# 4 Tips

- Sollte beim Compilieren mehr als ein Fehler auftreten, so sollte man sich zuerst den zuerst aufgelisteten beheben, und dann erneut compilieren, da viele der anderen Fehler bereits dadurch aufgehoben wurden.
- Besonders häufig treten Fehler auf, in denen der Compiler eine Methode oder eine Variable oder auch ein Attribut nicht finden kann. Man sollte dann zuerst auf Gross- und Kleinschreibung achten.
- Wenn man nicht genau weiss, wie ein Objekt oder eine Klasse benutzt werden soll, sollte man sich stehts die Java Dokumentation zur Hilfe nehmen. Vieles kann man auch in Büchern wie Go To Java 2 finden (Falls man nicht so gerne mit englischen Texten umgeht).
- Sollte bei einer Schleife oder einer if - Anweisung auch nur eine Anweisung folgen, so würde ich diese trotz allen auch in geschwungenen Klammern setzen, da das noch mal die Zugehörigkeit verdeutlicht. Ausserdem könnten später auch noch mehr Anweisungen hinzukommen.
- Sollte der Compiler mehrere Fehler anzeigen, so liegt es häufig auch an einer vergessenen Klammer oder an einem fehlenden Symikolon. Man sollte daher in der nähe der angegebenen Fehlerzeile danach suchen.

Ich wünsch euch viel Erfolg bei euren zukünftigen Programmen.
Thomas
Verbesserungsvorschläge bitte an thokrieg@agent.fho-emden.de.